# Angular

## Etape 1 : Setup

Install the CLI using the npm package manager:

```
npm install -g @angular/cli
```

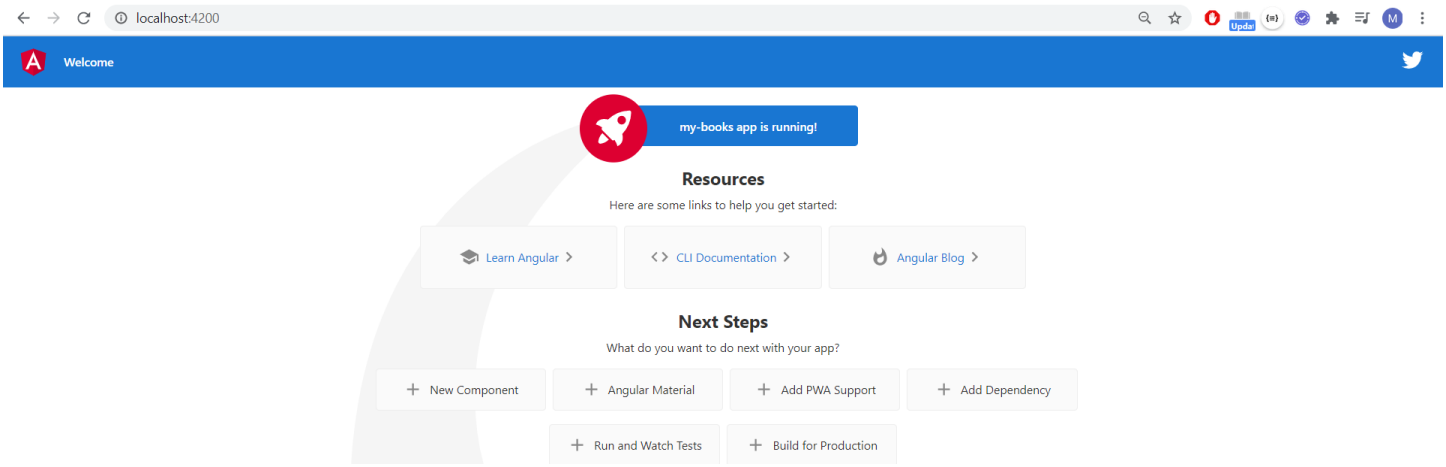
Créez un projet my-books

```
ng new my-books
```

Choisir : Routing
Choisir : CSS

### Test

```
cd my-books
ng serve
```



### Etape 2 : Création des composants

```
ng g c header --skipTests=true
```

```
D:\01_Tutos\01_DevOps\10_AWS\tps_new\06_SERVERLESS\python\cloud\my-books>ng g c header --skipTests=true
CREATE src/app/header/header.component.html (21 bytes)
CREATE src/app/header/header.component.ts (275 bytes)
CREATE src/app/header/header.component.css (0 bytes)
UPDATE src/app/app.module.ts (475 bytes)
```

```
ng g c home --skipTests=true
```

```
ng g c login --skipTests=true
```

```
ng g c register --skipTests=true
```

```
ng g c restapi --skipTests=true
```

### Etape 3 : Définir les routes d'une application

Éditer le fichier **app-routing.module.ts** pour configurer les routes

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

import {HomeComponent} from "./home/home.component";
import {RestapiComponent} from "./restapi/restapi.component";
import {LoginComponent} from "./login/login.component";
import {RegisterComponent} from "./register/register.component";

const routes: Routes = [
  {path: '', component: HomeComponent},
  {path: 'login', component: LoginComponent},
  {path: 'register', component: RegisterComponent},
  {path: 'restapi', component: RestapiComponent }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Lancer l'application et vérifier les urls suivants :

http://localhost:4200/
http://localhost:4200/login
http://localhost:4200/register
http://localhost:4200/restapi

## Etape 4 : Bootstrap et RxJs Setting

Entrez la commande suivante en ligne de commande

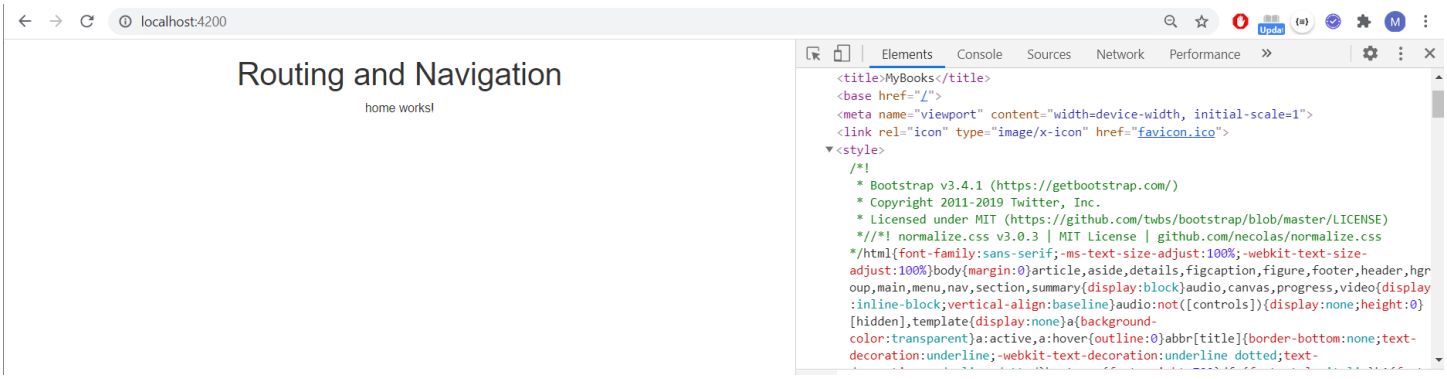
```
npm install --save bootstrap@3
```

Éditer le fichier **angular.json** et ajouter dans la section **styles**

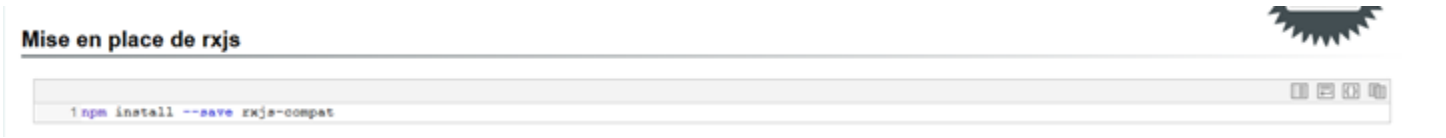
```
styles": [
    "node_modules/bootstrap/dist/css/bootstrap.min.css",
    "styles.css"
]
```

**! ! ! Relancer l'application**

Puis vérifier le chargement de Bootstrap dans l'outil de développement du navigateur **Chrome**




```
$ npm install rxjs
```



## Etape 5 : Adding navigation

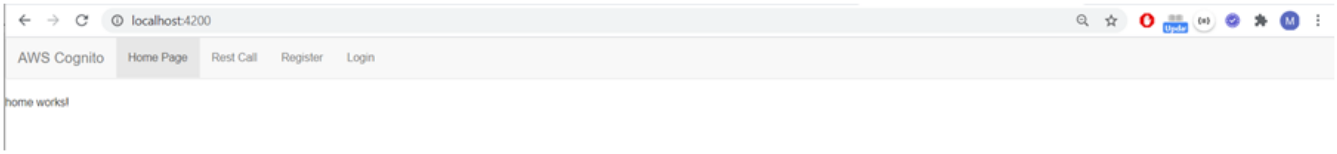
Éditer le fichier **header.component.html** et transformer

```
<nav class="navbar navbar-default">
 <div class="container-fluid">
  <div class="navbar-header">
   <a class="navbar-brand" href="#">AWS Cognito</a>
  </div>
  <ul class="nav navbar-nav">
   <li routerLinkActive="active" [routerLinkActiveOptions]="{exact: true}"><a routerLink="/">Home Page</a></li>
   <li routerLinkActive="active"><a  routerLink="/restapi">Rest Call</a></li>
   <li routerLinkActive="active"><a  routerLink="/register">Register</a></li>
   <li routerLinkActive="active"><a routerLink="/login">Login</a></li>
  </ul>
 </div>
</nav>
```

Éditer le fichier **app.component.html** et transformer

```
<app-header></app-header>
<router-outlet></router-outlet>
```

Tester




## Etape 6 : Activer ngModel pour le two way binding et du module HttpModule

Éditer le fichier **app.module.ts** et ajouter le code en vert.

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';

import { AppComponent } from './app.component';
import { ServerComponent } from './server/server.component';
import { ServersComponent } from './servers/servers.component';


@NgModule({
  declarations: [
    AppComponent,
    ServerComponent,
    ServersComponent
  ],
  imports: [
    BrowserModule,
    FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Faire la meme chose avec le module, `HttpClientModule` de **'@angular/common/http'**;

### Etape 7 : Création du modèle

Créez un répertoire **models** dans **my-books/src/app**.

Dans le dossier models, créer un fichier **book.model.ts**

```
export class Book {
    constructor(
        public bookId: string,
        public title: string,
        public year: number,
        public rating: number,
        public publisher: string
    ){}
}
```

### Etape 8 : Création du service Api

Générer un **service** nommé **Api**.

```
ng g s api/api --skipTests=true
```

Déclarer le service dans **app.module.ts**

```
import { BrowserModule } from '@angular/platform-browser';
...
import { ApiService } from './api/api.service';

@NgModule({
  declarations: [
    ...
  ],
  imports: [
    ...
  ],
  providers: [ApiService],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

## Etape 9 : Déclarer le chemin du WebService

### La bonne pratique pour déclarer le chemin principal

La bonne pratique pour déclarer le chemin principal consiste à pouvoir le faire à un endroit unique.

Sans intervention, il faudrait que la bonne valeur soit automatiquement prise en compte que l'on soit en développement ou en production.

Angular CLI propose une prise en charge de la gestion des environnements. A la génération du projet , 2 environnements avec leur fichier de configuration respectif ont été créés :

**src/environments/environment.ts** et **src/environments/environment.prod.ts**.

Nous allons utiliser ces fichiers pour définir l'url de notre api.

- Pour **src/environments/environment.ts**

  - Pour **src/environments/environment.ts**

```
export const environment = {
  production: false,

  //url vers l'api gateway de aws
  apiUrl: 'https://as1nv70iii.execute-api.eu-west-3.amazonaws.com/dev'
};
```

  - Pour **src/environments/environment.prod.ts.**

```
export const environment = {
  production: true,

  apiUrl: 'https://as1nv70iii.execute-api.eu-west-3.amazonaws.com/dev'
};
```

## Etape 10 : Codage du service

```
import { Book } from './../models/book.model';
import { environment } from './../../environments/environment';
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';
import { HttpClient, HttpHeaders, HttpParams } from '@angular/common/http';
import { catchError } from 'rxjs/operators';

//URL for api
const API_URL = environment.apiUrl;

@Injectable({
  providedIn: 'root'
})
export class ApiService {

    constructor(private http: HttpClient) { }

    createBook(book: Book): Observable<any> {
      let body = JSON.stringify(book);
      alert(book);
      const headers = new HttpHeaders().set("Content-Type", "application/json");
      return this.http.post(API_URL + '/books', body, { headers }).pipe(  catchError(this.handleError) );
    }

    getBooks(): Observable<Book[]>{
      const headers = new HttpHeaders().set("Content-Type", "application/json");
      return this.http.get<Book[]>(API_URL + '/books',{ headers }).pipe(  catchError(this.handleError) );
    }

    getBookById(bookId: string): Observable<Book> {
      const headers = new HttpHeaders().set("Content-Type", "application/json");
      return this.http.get<Book>(API_URL + '/books/' + bookId,{ headers }).pipe(catchError(this.handleError) );
    }

    updateBoock(book: Book): Observable<any> {
      const headers = new HttpHeaders().set("Content-Type", "application/json");
      return this.http.put(API_URL + '/books/' + book.bookId, book, { headers })
                  .pipe(  catchError(this.handleError) );
    }


    deleteBook(bookId: string): Observable<any> {
      const headers = new HttpHeaders().set("Content-Type", "application/json");
      return this.http.delete(API_URL + '/books/' + bookId, { headers }).pipe(catchError(this.handleError) );
    }

    private handleError(error: Response | any) {
      return Observable.throw(error);
    }
}
```

### Etape 11 : Mise à jour du composant RestApi

Editer le fichier **restapi.component.ts**

```
import { ApiService } from './../api/api.service';
import { Book } from './../models/book.model';
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-restapi',
  templateUrl: './restapi.component.html',
  styleUrls: ['./restapi.component.css']
})
export class RestapiComponent implements OnInit {

  constructor(private api:ApiService) { }

  ngOnInit(): void {
    this.getBooks();
  }

  _books : Book[] = [];

  // Get all books
  getBooks(){
    this.api.getBooks()
        .subscribe(
          (data) => { this._books = data}
        );
  }
}
```

Editer le fichier **restapi.component.html**

```
<style>
    table {
      border-collapse: separate;
      border-spacing: 50px 0;
    }

    td {
      padding: 10px 0;
    }
  </style>
  <section>
    <h2>Rest Api Output</h2>
    <table align="center">
      <thead>
        <tr>
          <th>Title</th>
          <th>Rating</th>
          <th>Year</th>
          <th>Publisher</th>
        </tr>
      </thead>
      <tbody>
        <tr *ngFor="let book of _books">
          <td>{{book.title}}</td>
          <td>{{book.rating}}</td>
          <td>{{book.year}}</td>
          <td>{{book.publisher}}</td>
        </tr>
      </tbody>
    </table>
  </section>
```

## Etape 12 : Test